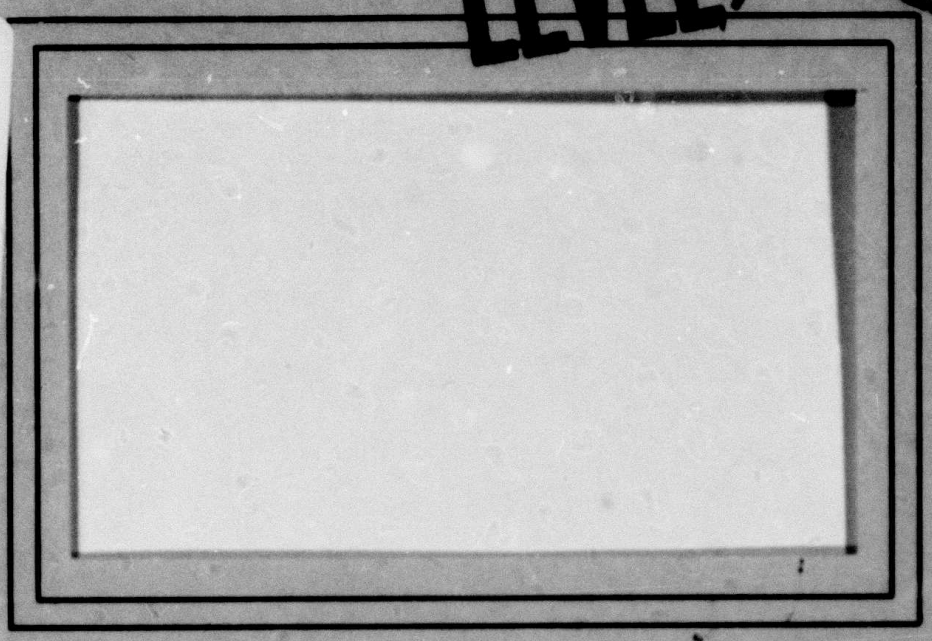





# UNIVERSITY OF MARYLAND
# COMPUTER SCIENCE CENTER

COLLEGE PARK, MARYLAND
20742

(15) DAAG53-76-C-0138, DARPA Order-3206

(9) Technical rept.,

(14) TR-770
DAAG-53-76C-0138

(11) May 79

(12) 17

(6) LINEAR TIME CALCULATIONS
OF GEOMETRIC PROPERTIES USING QUADTREES

(10) Michael Shneier
Computer Science Center
University of Maryland
College Park, MD 20742



ABSTRACT

This paper describes algorithms for computing geometric properties of binary images represented as quadtrees. All the algorithms involve a simple traversal of the tree. Each algorithm, however, performs different operations at the nodes of the tree. Algorithms are presented for finding the area, centroid, union, intersection, and complement of binary images. All the algorithms are linear in the number(s) of nodes in the tree(s).



---

The support of the Defense Advanced Research Projects Agency and the U.S. Army Night Vision Laboratory under Contract DAAG-53-76C-0138 (DARPA Order 3206) is gratefully acknowledged, as is the help of Kathryn Riley in preparing this paper. The paper has benefitted greatly from discussions with Chuck Dyer, Azriel Rosenfeld, and Hanan Samet.

403 018

JOB

## 1. Introduction

Shape representation is an important aspect of image processing. It is useful to have a representation that allows easy calculation of shape features and shape differences. This paper presents several algorithms that operate on regions represented by quadtrees [1]. In particular, a linear time method of calculating the moments of a region is presented, as well as algorithms for the operations of union, intersection, and complementation of quadtrees.

It is assumed that the image is a square binary $2^n x 2^n$ array composed of black and white points. The quadtree representation of such an image is obtained by its successive subdivision into quadrants. The root node of the tree represents the whole image. If this is not all black or all white, it is split into four quadrants, represented by four sons of the root node. Again, each of these sons that does not represent part of the image that is all white or all black is subdivided into four sons. Eventually, this process terminates, at worst with individual pixels as leaves in the tree.

## 2. Definitions

Each node in the quadtree holds six pieces of information. The first five are pointers to the node's father and to its four sons. (The root node has no father, and leaf nodes have no sons.) The sixth piece of information describes the node itself. It has value BLACK if the region represented by the node is all black, WHITE if it is all white, and GRAY otherwise. GRAY nodes represent non-terminals in the quadtree, while BLACK and WHITE nodes are leaf nodes.

The four sons of a node are labeled NW (northwest), NE (northeast), SW (southwest), and SE (southeast). Notice that if a node is of size $2^n x 2^n$ (i.e., it corresponds to a region of $2^n x 2^n$ pixels in the image), its sons will each be of size $2^{n-1} x 2^{n-1}$.

The square array of pixels on which the quadtree is defined has the following coordinate system imposed on it.

The leftmost, topmost pixel is at the (0,0) coordinate. The rightmost, bottommost pixel is at the $(2^n-1, 2^n-1)$ coordinate. The coordinates (x,y) are such that the x-value increases from left to right (west to east), and the y coordinate increases form top to bottom (north to south) in the image. The particular coordinate system chosen is not crucial; this one makes the calculations particularly easy.

## 3. The algorithms

All the algorithms to be presented here are based on a very simple tree-traversal algorithm. In order to compute moments, it is necessary to visit every BLACK leaf in the tree and calculate some function on the basis of the distributions of the leaves. For the set operations, comparisons between the structure of two quadtrees require parallel traversal of the trees.

Where the quadtree representation scores over alternative array-based representations is in its ability to process large blocks of the image at once, when these blocks form single leaves in the quadtree. Samet [2] has shown that the worst case size of a quadtree is 4Bn, where B is the number of BLACK nodes and the image is of size $2^n$ by $2^n$. The expected size should be much smaller, so that substantial processing gains may be achieved.

The tree traversal algorithm uses a form of postorder [3]. The procedure is to traverse the NW subtree, traverse the NE subtree, traverse the SW subtree, traverse the SE subtree, and then visit the node. It should be noted that the particular traversal order chosen is not crucial for the set-theoretic algorithms. Equivalent algorithms could be devised for the other orders.

### A1: Area

This algorithm finds the area of an image represented by a quadtree, where area is defined as the total number of black

pixels in the image.

Input to the algorithm is a pointer to the root node of a quadtree, and a number n denoting the log of the diameter of the image (i.e., the image is of size $2^n x 2^n$).

```
integer procedure AREA(QUADTREE,N);
    /* find the number of black pixels in an image of size
       2^N by 2^N represented by a quadtree */
begin
    node QUADTREE;
    integer BLACKAREA;
    level N;
    quadrant I;
    BLACKAREA:=0;
    if GRAY(QUADTREE) then
        for I in (NW,NE,SW,SE) do
            BLACKAREA:=BLACKAREA+
                AREA(SON(QUADTREE,I),N-1);
    else if BLACK(QUADTREE) then
        BLACKAREA:=BLACKAREA+2↑(2*N);
    return(BLACKAREA);
end;
```

It can easily be seen that this algorithm vists every black leaf once and only once, thus giving the correct area.

Algorithms for finding higher moments are complicated by the necessity of relating the position of a node in the quadtree to the coordinates of a region in the image. Each quadrant is specified by the coordinates of its top left

corner, and by its size. We give an algorithm for finding the centroid of an image. The other moments are computed similarly.

A2: Centroid

The centroid of a binary image is a point $(\bar{x},\bar{y})$ such that $\bar{x}$ is the weighted average value of the x-coordinates of all the black points of the image and $\bar{y}$ is the weighted average of the y-coordinates of the black points. In other words, if there are m black points in the image, $(x_1,y_1),\ldots,(x_m,y_m)$, the centroid is $(\bar{x},\bar{y}) = (\frac{\Sigma x_i}{m},\frac{\Sigma y_i}{m})$ .

The centroid procedure is called with a pointer to the root node of the quadtree as the QUADTREE parameter. N is initially equal to n, where the input image is size $2^n$ by $2^n$; all the other parameters have value 0.

```
procedure CENTROID(QUADTREE,N,XCOORD,YCOORD,X,Y,MASS,XCENT,YCENT);
/* calculate the centroid of QUADTREE for an image of size 2^N x 2^N.
   XCOORD, YCOORD are the coordinates of the top
   left corner of the tree.
   X,Y are the coordinates of the centroid.
   MASS is the number of points.
   XCENT and YCENT are the centroid values. */
begin
    node QUADTREE;
    coord X,Y,XCOORD,YCOORD,X1,Y1,XCENT,YCENT;
    level N;
    integer MASS,M,J,K,M1;
    quadrant I;
    X:=0; Y:=0; MASS:=0;
```

```
    if GRAY(QUADTREE) then
        for (I,J,K) in {(NW,0,0),(NE,1,0),(SW,0,1),(SE,1,1)} do
            begin
            CENTROID(SON(QUADTREE,I),N-1,XCOORD+J*2↑(N-1),
                YCOORD+K*2↑(N-1),X1,Y1,M1,XCENT,YCENT);
            X:=X+X1;
            Y:=Y+Y1;
            MASS:=MASS+M1;
            end;
    else if BLACK(QUADTREE) then
        begin
        X:=XCOORD+2↑(N-1);
        Y:=YCOORD+2↑(N-1);
        MASS:=2↑(N+1)
        end;
    if MASS=0 then
        begin
        XCENT:=0;
        YCENT:=0;
        end;
    else begin
        XCENT:=X/MASS;
        YCENT:=Y/MASS;
        end;
    return;
end;
```

The main difference between the centroid and area algorithms is in the calculation of the coordinates. This is a simple process, and does not affect the rest of the algorithm. It is easy to see, once again, that each BLACK leaf in the tree is visited once and only once.

The other moments can be calculated in an analogous way.

Set Operations

It is often desired to compare two regions in an image and to find what is common to them. This involves intersecting the two regions and determining what points are common to both of them. Other set operations that are useful are complementation and union. The algorithms for these operations involve, once again, a tree traversal. However, in the union and intersection algorithms, it may not be necessary to traverse the whole tree. The union and intersection algorithms are special cases of the superposition algorithm of Hunter and Steiglitz [4]. They are included because of their simplicity and their similarity to the other algorithms.

A3:Complement

Constructing the complement of an image involves changing black pixels into white, and white pixels or nodes to black. This is a very simple operation in a quadtree, and does not change the structure of the tree at all.

Input to the procedure is a pointer to the root of the tree.

```
procedure  COMPLEMENT(QUADTREE);
/* change a quadtree into its complement */
begin
node QUADTREE;
quadrant I;
if GRAY(QUADTREE) then
    for I in {NW,NE,SW,SE} do
        COMPLEMENT(SON(QUADTREE,I));
else if BLACK(QUADTREE) then
    TYPE(QUADTREE):=WHITE;
     else /* a WHITE node */
     TYPE(QUADTREE):=BLACK;
end;
```

A4:Intersection

This procedure finds the logical AND of two binary images represented by quadtrees. The algorithm involves traversing the trees in parallel. When one tree has a son that is a BLACK leaf, while the other has a corresponding son that is not BLACK, the BLACK leaf is replaced by the corresponding subtree. If one tree has a leaf that is WHITE, the intersection tree will have a corresponding WHITE leaf. Finally, if both trees have GRAY nodes in corresponding positions, the nodes' sons are examined recursively, using the same process.

Input to the procedure is a pointer to the root of each tree.

```
quadtree procedure INTERSECTION(TREE1,TREE2);
/* returns the intersection of TREE1 and TREE2 */
begin
    node TREE1,TREE2,INTERSECT;
    quadrant I;
    if BLACK(TREE1) or WHITE(TREE2) then
        return(COPY(TREE2));
    else if BLACK(TREE2) or WHITE(TREE1) then
        return(COPY(TREE1));
    INTERSECT:=CREATENODE();  /*create a root node */
    for I in {NW,NE,SW,SE} do
        begin
            SON(INTERSECT,I):=INTERSECTION(SON(TREE1,I),SON(TREE2,I));
            FATHER(SON(INTERSECT,I)):=INTERSECT;
        end;
    return(INTERSECT);
end;
```

```
quadtree procedure COPY(TREE);
/* creates a tree structure identical to TREE */
begin
    quadtree TREE,NEWTREE;
    quadrant I;
    NEWTREE:=CREATENODE();
    /* create a node with NULL FATHER, SON, and TYPE nodes */
    TYPE(NEWTREE):=TYPE(TREE);
    for I in {NW,NE,SW,SE,} do
        if SON(TREE,I)=NULL then SON(NEWTREE,I):=NULL;
        else begin
            SON(NEWTREE,I):=COPY(SON(TREE,I));
            FATHER(SON(NEWTREE,I)):=NEWTREE;
            end;
    return(NEWTREE);
end;
```

A5:Union

The union algorithm is very similar to the intersection algorithm. The trees are again traversed in parallel, and decisions are made whenever either of the traversals reaches a leaf. The decisions are the mirror images of those in the intersection algorithm: when a BLACK leaf is encountered, this becomes the subtree at the current position in the union tree. A WHITE leaf in one tree results in the corresponding subtree of the other tree becoming the subtree at the current position of the union tree. When there are two GRAY nodes, the procedure is called recursively for the subtrees rooted at these nodes.

Input to the algorithm is a pointer to the root node of each tree.

```
quadtree procedure UNION(TREE1,TREE2);
/* construct the union of TREE1 and TREE2 */
begin
    node TREE1,TREE2,UNI;
    quadrant I;
    if BLACK(TREE2) or WHITE(TREE1) then
        return(COPY(TREE2))
    else if WHITE(TREE2) or BLACK(TREE1) then
        return(COPY(TREE1));
    UNI:=CREATENODE(); /* create the root node */
    for I in {NW,NE,SW,SE} do
        begin
            SON(UNI,I):=UNION(SON(TREE1,I),
                              SON(TREE2,I));
            FATHER(SON(UNI,I)):=UNI;
        end;
    return(UNI);
end;
```

Both the union and intersection algorithms can be generalized to deal with an arbitrary number of trees. In all cases, the intersection algorithm takes time proportional to the traversal time of the smallest tree, and the union algorithm takes time proportional to that of the second largest tree. All the algorithms presented have times linear in the sizes of the trees on which they operate.

## References


[1] A. Klinger and C. R. Dyer. Experiments in picture representation using regular decomposition. Computer Graphics and Image Processing 5, 1976, 68-105.

[2] H. Samet. Computing perimeters of images represented by quadtrees. Computer Science TR-755, University of Maryland, College Park, Maryland, April 1979.

[3] D. E. Knuth. The Art of Computer Programming, Vol. 1: Fundamental Algorithms. Addison-Wesley, Reading, Mass., 1973.

[4] G. M. Hunter and K. Steiglitz. Operations on images using quadtrees. IEEE Trans. PAMI 1, 1979, 145-153.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

| REPORT DOCUMENTATION PAGE | | READ INSTRUCTIONS BEFORE COMPLETING FORM |
|---|---|---|
| 1. REPORT NUMBER | 2. GOVT ACCESSION NO. | 3. RECIPIENT'S CATALOG NUMBER |
| 4. TITLE (and Subtitle)<br>LINEAR TIME CALCULATIONS OF GEOMETRIC PROPERTIES USING QUADTREES | | 5. TYPE OF REPORT & PERIOD COVERED<br>Technical<br>6. PERFORMING ORG. REPORT NUMBER<br>TR-770 |
| 7. AUTHOR(s)<br>Michael Shneier | | 8. CONTRACT OR GRANT NUMBER(s)<br>DAAG-53-76C-0138 |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS<br>Computer Science Center<br>University of Maryland<br>College Park, MD 20742 | | 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS |
| 11. CONTROLLING OFFICE NAME AND ADDRESS<br>U.S. Army Night Vision Laboratory<br>Fort Belvoir, VA 22060 | | 12. REPORT DATE<br>May 1979<br>13. NUMBER OF PAGES<br>15 |
| 14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) | | 15. SECURITY CLASS. (of this report)<br>Unclassified<br>15a. DECLASSIFICATION/DOWNGRADING SCHEDULE |

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

| | |
|---|---|
| Pattern recognition | Centroid |
| Image processing | Boolean operations |
| Moments | Binary images |
| Area | Quadtrees |

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This paper describes algorithms for computing geometric properties of binary images represented as quadtrees. All the algorithms involve a simple traversal of the tree. Each algorithm, however, performs different operations at the nodes of the tree. Algorithms are presented for finding the area, centroid, union, intersection, and complement of binary images. All the algorithms are linear in the number(s) of nodes in the tree(s).

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)